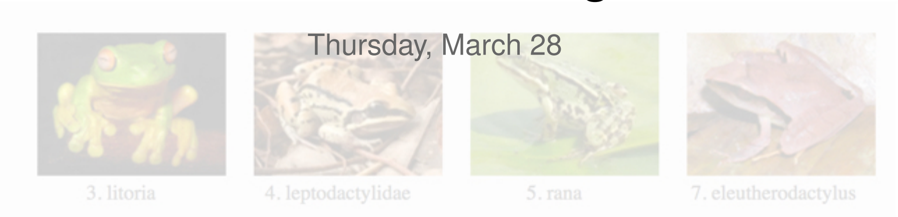
S&DS 171

**YData: Text Data Science**

# Word Embeddings

Thursday, March 28



# For Today

- Class-based LMs
- Embeddings
- Visualization
- Starting the Lab

# Class-based bigram model

- Model takes form

$$
\begin{aligned}
p(w_2 \mid w_1) &= p(\mathrm{class}(w_2) \mid \mathrm{class}(w_1))\, p(w_2 \mid \mathrm{class}(w_2)) \\
&= p(c_2 \mid c_1)\, p(w_2 \mid c_2)
\end{aligned}
$$

Brown et al., "Class-based *n*-gram models of natural language"

# Class-based bigram model

- Model takes form

$$\begin{aligned} p(w_2 \mid w_1) &= p(\mathrm{class}(w_2) \mid \mathrm{class}(w_1))\, p(w_2 \mid \mathrm{class}(w_2)) \\ &= p(c_2 \mid c_1)\, p(w_2 \mid c_2) \end{aligned}$$

- Use bottom-up agglomerative clustering to group the words.

---

Brown et al., "Class-based *n*-gram models of natural language"

# Class-based bigram model

- Model takes form

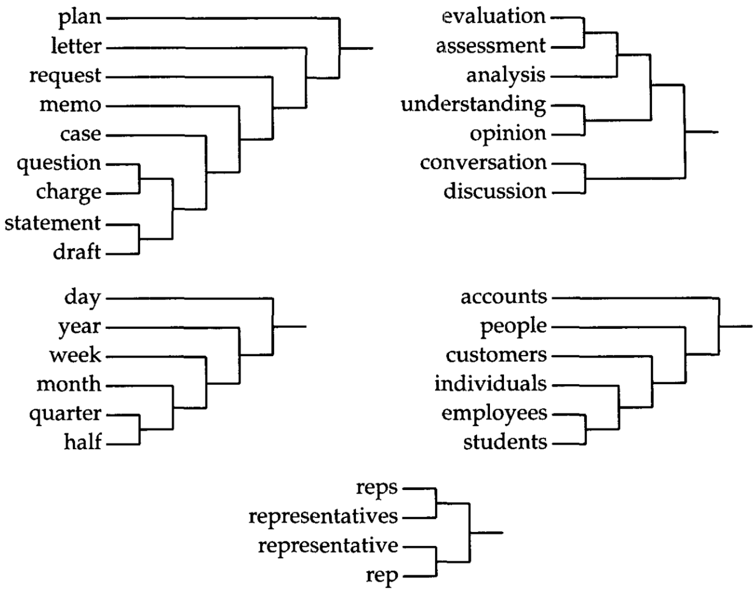
$$
\begin{aligned}
p(w_2 \mid w_1) &= p(\text{class}(w_2) \mid \text{class}(w_1))\, p(w_2 \mid \text{class}(w_2)) \\
&= p(c_2 \mid c_1)\, p(w_2 \mid c_2)
\end{aligned}
$$

- Use bottom-up agglomerative clustering to group the words.
- In each step, merge the pair of classes that gives the smallest reduction in likelihood of the data.

---

Brown et al., "Class-based *n*-gram models of natural language"

# Sample merges



# Sample clusters

Friday Monday Thursday Wednesday Tuesday Saturday Sunday weekends Sundays Saturdays

June March July April January December October November September August

people guys folks fellows CEOs chaps doubters commies unfortunates blokes

down backwards ashore sideways southward northward overboard aloft downwards adrift

water gas coal liquid acid sand carbon steam shale iron

great big vast sudden mere sheer gigantic lifelong scant colossal

man woman boy girl lawyer doctor guy farmer teacher citizen

American Indian European Japanese German African Catholic Israeli Italian Arab

pressure temperature permeability density porosity stress velocity viscosity gravity tension

mother wife father son husband brother daughter sister boss uncle

machine device controller processor CPU printer spindle subsystem compiler plotter

John George James Bob Robert Paul William Jim David Mike

anyone someone anybody somebody

feet miles pounds degrees inches barrels tons acres meters bytes

director chief professor commissioner commander treasurer founder superintendent dean custodian

liberal conservative parliamentary royal progressive Tory provisional separatist federalist PQ

had hadn't hath would've could've should've must've might've

asking telling wondering instructing informing kidding reminding bothering thanking deposing

that tha theat

head body hands eyes voice arm seat eye hair mouth

# Group globally, compute locally

- Clusters contain syntactic and semantic elements
- Surprising, since use local statistics only
- “A word is known by the company it keeps”

# Information theoretic view

A clustering corresponds to a partition $\pi$ of the word vocabulary. Under the class bigram model, the log-likelihood of the data can be shown to satisfy

$$\ell(\pi) = \text{constant} + I(C_1, C_2)$$

where $I$ is mutual information.

We want the identity of the previous class (cluster) to contain a lot of information about the identity of the next class.

# Pointwise mutual information (PMI)

Average mutual information

$$I(W_1, W_2) = \sum_{w_1, w_2} p(w_1, w_2) \log \frac{p(w_1, w_2)}{p(w_1)p(w_2)}$$

Related statistic is "pointwise mutual information" (PMI)

$$\log \left( \frac{p_{\text{near}}(w_1, w_2)}{p(w_1)p(w_2)} \right)$$

- How likely are specific words/clusters to co-occur together within some window, compared to if they were independent?

# Example clusters from PMI

we our us ourselves ours
question questions asking answer answers answering
performance performed perform performs performing
tie jacket suit
write writes writing written wrote pen
morning noon evening night nights midnight bed
attorney counsel trial court judge
problems problem solution solve analyzed solved solving
letter addressed enclosed letters correspondence
large size small larger smaller
operations operations operating operate operated
school classroom teaching grade math
street block avenue corner blocks
table tables dining chairs plate
published publication author publish writer titled
wall ceiling walls enclosure roof
sell buy selling buying sold

# Shortcomings of word clusters

- No centroid in $\mathbb{R}^d$, or Euclidean structure
- Can't use vector space operations
- One-hot representation wasteful
- These are addressed with "distributed representations" (next)

# Core idea of embeddings

- Combine this type of representation / parameterization with PMI-like scores to get embedding vectors.
- Can be applied whenever have cooccurrence data.

# Embedding language model

Replacing classes with embedding vectors gives

$$p(w_2 \mid w_1) = \frac{\exp(\phi(w_2)^T \phi(w_1)}{\sum_w \exp(\phi(w)^T \phi(w_1))}.$$

A calculation shows that

$$\begin{aligned}\ell(\phi) &= \sum_{w_1, w_2} p(w_1, w_2) \log p(\phi_2 \mid \phi_1) p(w_2 \mid \phi_2) \\ &= I(\Phi_1, \Phi_2) + \text{constant}\end{aligned}$$

Thus, we want embedding vectors with high mutual information.

# Constructing embeddings

Carry out stochastic gradient descent over the embedding vectors $\phi \in \mathbb{R}^d$ (where $d \approx 50$–100 is chosen by trial and error)

This is what Mikolov et al. (2014, 2015) did at Google. With a couple of heuristics:

“Distributed representations of words,” (2014) “Efficient representations of words in vector space” (2015)

# Constructing embeddings

Heuristics used:

- Skip-gram: predict surrounding words from current word

“Distributed representations of words,” (2014) “Efficient representations of words in vector space” (2015)

# Constructing embeddings

Heuristics used:

- Skip-gram: predict surrounding words from current word
- An issue with this is that it "over generates" the data. With text `the lazy brown fox jumped` we will have $p(\texttt{brown}|\texttt{lazy})$ and $p(\texttt{brown}|\texttt{fox})$

"Distributed representations of words," (2014) "Efficient representations of words in vector space" (2015)

# Constructing embeddings

Heuristics used:

- Skip-gram: predict surrounding words from current word
- An issue with this is that it "over generates" the data. With text `the lazy brown fox jumped` we will have $p(\texttt{brown}|\texttt{lazy})$ and $p(\texttt{brown}|\texttt{fox})$
- Second is computational. The bottleneck is computing the denominator in the logistic (softmax) probability.

"Distributed representations of words," (2014) "Efficient representations of words in vector space" (2015)

# Constructing embeddings

Heuristics used:

- Skip-gram: predict surrounding words from current word
- An issue with this is that it "over generates" the data. With text `the lazy brown fox jumped` we will have $p(\texttt{brown}|\texttt{lazy})$ and $p(\texttt{brown}|\texttt{fox})$
- Second is computational. The bottleneck is computing the denominator in the logistic (softmax) probability.
- Use "negative sampling"

---

"Distributed representations of words," (2014) "Efficient representations of words in vector space" (2015)

# Analogies

These heuristics enable training on very large text collections. Leads to vector representations of words with interesting properties.

For example, analogies:

`king` is to `man` as ? is to `woman`

# Analogies

These heuristics enable training on very large text collections. Leads to vector representations of words with interesting properties.

For example, analogies:

`king` is to `man` as ? is to `woman`

`Paris` is to `France` as ? is to `Germany`

# Analogies

These heuristics enable training on very large text collections. Leads to vector representations of words with interesting properties.

For example, analogies:

$\texttt{king}$ is to $\texttt{man}$ as ? is to $\texttt{woman}$

$\texttt{Paris}$ is to $\texttt{France}$ as ? is to $\texttt{Germany}$

$$\phi(\texttt{king}) - \phi(\texttt{man}) \overset{?}{\approx} \phi(\texttt{queen}) - \phi(\texttt{woman})$$

$$\widehat{w} = \arg\min_{w} \|\phi(\texttt{king}) - \phi(\texttt{man}) + \phi(\texttt{woman}) - \phi(w)\|^2$$

Does $\widehat{w} = \texttt{queen}$?

# Learned Analogies

Table 8: *Examples of the word pair relationships, using the best word vectors from Table 4 (Skip-gram model trained on 783M words with 300 dimensionality).*

| Relationship | Example 1 | Example 2 | Example 3 |
|---|---|---|---|
| France - Paris | Italy: Rome | Japan: Tokyo | Florida: Tallahassee |
| big - bigger | small: larger | cold: colder | quick: quicker |
| Miami - Florida | Baltimore: Maryland | Dallas: Texas | Kona: Hawaii |
| Einstein - scientist | Messi: midfielder | Mozart: violinist | Picasso: painter |
| Sarkozy - France | Berlusconi: Italy | Merkel: Germany | Koizumi: Japan |
| copper - Cu | zinc: Zn | gold: Au | uranium: plutonium |
| Berlusconi - Silvio | Sarkozy: Nicolas | Putin: Medvedev | Obama: Barack |
| Microsoft - Windows | Google: Android | IBM: Linux | Apple: iPhone |
| Microsoft - Ballmer | Google: Yahoo | IBM: McNealy | Apple: Jobs |
| Japan - sushi | Germany: bratwurst | France: tapas | USA: pizza |

Mikolov et al., “Distributed representations of words,” (2014); “Efficient representations of words in vector space” (2015)

# Evaluation Analogies

| Type of relationship | Word Pair 1 | | Word Pair 2 | |
|---|---|---|---|---|
| Common capital city | Athens | Greece | Oslo | Norway |
| All capital cities | Astana | Kazakhstan | Harare | Zimbabwe |
| Currency | Angola | kwanza | Iran | rial |
| City-in-state | Chicago | Illinois | Stockton | California |
| Man-Woman | brother | sister | grandson | granddaughter |
| Adjective to adverb | apparent | apparently | rapid | rapidly |
| Opposite | possibly | impossibly | ethical | unethical |
| Comparative | great | greater | tough | tougher |
| Superlative | easy | easiest | lucky | luckiest |
| Present Participle | think | thinking | read | reading |
| Nationality adjective | Switzerland | Swiss | Cambodia | Cambodian |
| Past tense | walking | walked | swimming | swam |
| Plural nouns | mouse | mice | dollar | dollars |
| Plural verbs | work | works | speak | speaks |

Mikolov et al., “Distributed representations of words,” (2014); “Efficient representations of words in vector space” (2015)

# GloVe

Shortly after: Stanford group introduced a computational expedient (with attempt to give a "principled" motivation)

$$\mathcal{O}(\phi) = \sum_{w_1, w_2} f(c_{w_1,w_2}) \left(\phi(w_1)^T \phi(w_2) - \log c_{w_1,w_2}\right)^2$$

where $c_{w,w'}$ are cooccurrence counts.

- A type of regression estimator. Can interpret/relate this to other objectives.
- Main advantage is that SGD can be carried out much more efficiently

Pennington et al., "GloVe: Global vectors for word representation," (2015)

# GloVe

$$\mathcal{O}(\phi) = \sum_{w_1, w_2} f(c_{w_1,w_2}) \left( \phi(w_1)^T \phi(w_2) - \log c_{w_1,w_2} \right)^2$$

where $c_{w,w'}$ are cooccurrence counts.

- Heuristic weighting function

$$f(x) = \left( \frac{x}{x_{\max}} \right)^{\alpha}$$

  where $\alpha = 3/4$ set empirically.

- So $10^{-4} \mapsto 10^{-3}$. Each order of magnitude down gets "boosted" by 1/4-magnitude.

---

Pennington et al., "GloVe: Global vectors for word representation," (2015)

# GloVe site and code

## GloVe: Global Vectors for Word Representation

Jeffrey Pennington, Richard Socher, Christopher D. Manning

**Introduction**

GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space.

**Getting started (Code download)**

- Download the code (licensed under the Apache License, Version 2.0)
- Unpack the files: unzip GloVe-1.2.zip
- Compile the source: cd GloVe-1.2 && make
- Run the demo script: ./demo.sh
- Consult the included README for further usage details, or ask a question
- The code is also available on GitHub

**Download pre-trained word vectors**

- Pre-trained word vectors. This data is made available under the Public Domain Dedication and License v1.0 whose full text can be found at: http://www.opendatacommons.org/licenses/pddl/1.0/.
  - Wikipedia 2014 + Gigaword 5 (6B tokens, 400K vocab, uncased, 50d, 100d, 200d, & 300d vectors, 822 MB download): glove.6B.zip
  - Common Crawl (42B tokens, 1.9M vocab, uncased, 300d vectors, 1.75 GB download): glove.42B.300d.zip
  - Common Crawl (840B tokens, 2.2M vocab, cased, 300d vectors, 2.03 GB download): glove.840B.300d.zip
  - Twitter (2B tweets, 27B tokens, 1.2M vocab, uncased, 25d, 50d, 100d, & 200d vectors, 1.42 GB download): glove.twitter.27B.zip
- Ruby script for preprocessing Twitter data

**Citing GloVe**

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. [pdf] [bib]

**Highlights**

1. **Nearest neighbors**

   The Euclidean distance (or cosine similarity) between two word vectors provides an effective method for measuring the linguistic or semantic similarity of the corresponding words. Sometimes, the nearest neighbors according to this metric reveal rare but relevant words that lie outside an average human's vocabulary. For example, here are the closest words to the target word *frog*:

   0. *frog*
   1. frogs
   2. toad
   3. litoria
   4. leptodactylidae
   5. rana
   6. lizard
   7. eleutherodactylus

3. litoria

4. leptodactylidae

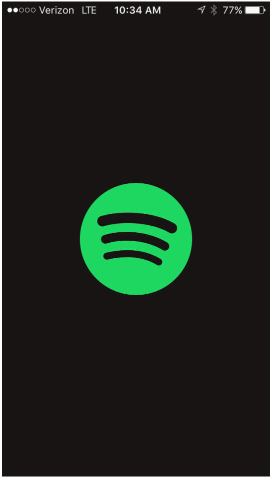
5. rana

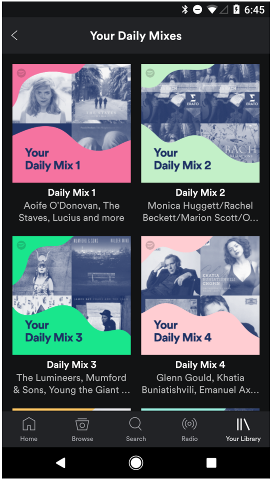
7. eleutherodactylus

2. **Linear substructures**

   The similarity metrics used for nearest neighbor evaluations produce a single scalar that quantifies the relatedness of two words. This simplicity can be problematic since two given words almost always exhibit more intricate relationships than can be captured by a single number. For example, *man* may be regarded as similar to *woman* in that both words describe human beings; on the other hand, the two words are often considered opposites since they highlight a primary axis along which humans differ from one another.

# Recommendation via Embedding



# Embedding embeddings: t-SNE

- How can we visualize the embeddings?

# Embedding embeddings: t-SNE

- How can we visualize the embeddings?
- We're in a very high dimensional space

# Embedding embeddings: t-SNE

- How can we visualize the embeddings?
- We're in a very high dimensional space
- Many visualization techniques exist. A currently popular one is t-SNE: "Student-t Stochastic Neighborhood Embedding"

# Embedding embeddings: t-SNE

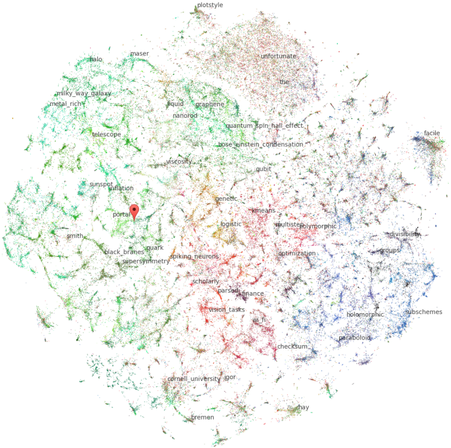
- How can we visualize the embeddings?
- We're in a very high dimensional space
- Many visualization techniques exist. A currently popular one is t-SNE: "Student-t Stochastic Neighborhood Embedding"
- Intuition: If $\phi(w_i)$ is very close to $\phi(w_j)$ then $y_i$ will be close to $y_j$. (long distances may be stretched further...)

# t-SNE: More info and examples

`https://lvdmaaten.github.io/tsne/`
`http://cs.stanford.edu/people/karpathy/tsnejs/`

Note: This is just a visualization technique, to give intuition for the high dimensional embedding

# t-SNE: Examples




http://www.cs.cornell.edu/~ginsparg/arxiv/gmaps2.html

# Summary: Word embeddings

- Word embeddings are vector representations of words, learned from cooccurrence statistics
- The models can be viewed in terms of class-based bigram models
- Surprising semantic relations are encoded in linear relations
- Various heuristics have been introduced to get scalability
- Embeddings improve with more data
- t-SNE is an algorithm for visualizing embeddings